Projective Integration for Moment Equations

Julian Koellermeier

Peking University

November 21st, 2018
Today’s Topic

- Quadrature-Based Moment Equations
- Filtered Hyperbolic Moment Equations
- Projective Integration for Moment Equations
- Hyperbolic Shallow Water Moment Equations
- Adaptive Moment Model
Today’s Topic

- Quadrature-Based Moment Equations
- Filtered Hyperbolic Moment Equations
- Projective Integration for Moment Equations
- Hyperbolic Shallow Water Moment Equations
- Adaptive Moment Model
1 Introduction
   - Stiff problems
   - Runge-Kutta Schemes

2 Projective Integration for Moment Equations
   - Definition
   - Stability
   - Telescopic PI

3 Summary
   - Summary
Consider the following PDE

\[ \partial_t u + \partial_x F(u) = -\frac{1}{\epsilon} S(u), \quad x \in \Omega, \ t \in \mathbb{R}^+ \]

\[ \Rightarrow \quad \partial_t u = -\frac{1}{\epsilon} S(u) - \partial_x F(u) \]

Spatial discretization \( x_1, x_2, \ldots, x_N \), for \( N \in \mathbb{N} \) using \( u = (u_1, u_2, \ldots, u_N) \)

\[ \partial_t u = D_t(u) \]
Stiff PDEs/ODEs

Consider the following PDE

\[ \partial_t u + \partial_x F(u) = -\frac{1}{\epsilon} S(u), \quad x \in \Omega, \ t \in \mathbb{R}^+ \]

⇒ \[ \partial_t u = -\frac{1}{\epsilon} S(u) - \partial_x F(u) \]

Spatial discretization \( x_1, x_2, \ldots, x_N \), for \( N \in \mathbb{N} \) using \( u = (u_1, u_2, \ldots, u_N) \)

\[ \partial_t u = D_t (u) \]

In this talk I want to discuss a numerical method for stiff problems
Consider the following system for system matrix $A \in \mathbb{R}^{M \times M}$ and force $f \in \mathbb{R}^{M}$

$$\partial_t u = Au + f(t)$$

The solution is given by

$$y(t) = \sum_{i=1}^{M} c_i \exp(\lambda_i t) v_i + g(t)$$

for $c_i = \text{const}$, eigenvalues $\lambda_i$, eigenvectors $v_i$ and some stationary solution $g(t)$.

The solution will approach the steady state solution as $t \to \infty$. 
Definition of stiff problem

Consider the following system for system matrix $A \in \mathbb{R}^{M \times M}$ and force $f \in \mathbb{R}^{M}$

$$\partial_t u = Au + f(t)$$

The solution is given by

$$y(t) = \sum_{i=1}^{M} c_i \exp(\lambda_i t)v_i + g(t)$$

for $c_i = \text{const}$, eigenvalues $\lambda_i$, eigenvectors $v_i$ and some stationary solution $g(t)$.

The solution will approach the steady state solution as $t \to \infty$.

Decay of each component might be different, depending on $\text{Re}(\lambda_i)$. 
Definition of stiff problem

Define the stiffness ratio

\[
\frac{\max |Re(\lambda_i)|}{\min |Re(\lambda_i)|}
\]

Large stiffness ratio can lead to stability problems.

Stiffness property

- large stiffness ratio
- some components of the solution decay much faster than others
- stability limits time step size \( \Delta t \) instead of accuracy
Examples for stiff ODEs

- Chemical reactions with different reaction speeds
  \[ A + X \xrightarrow{k_1} 2X \]
  \[ X + Y \xrightarrow{k_2} 2 \]
  \[ Y \xrightarrow{k_3} B \]

- Combustion problems

- Fluid dynamics including sonic waves versus bulk movement of flow

- Kinetic equation
  \[ \partial_t u + A_M \partial_x u = -\frac{1}{\epsilon} S(u) \]
Explicit vs Implicit schemes

\[ u^{n+1} = u^n + \Delta t \cdot D_t(u^n) \]

Explicit schemes, e.g. Forward Euler (FE)

- explicit update formula
- straightforward implementation
- restrictive time step constraint (no A-stability)

\[ u^{n+1} = u^n + \Delta t \cdot D_t(u^{n+1}) \]

Implicit schemes, e.g. Implicit Euler (IE)

- require solution of (non-)linear system ⇒ slow
- more difficult to implement
- no time step constraint (A-stability)
Example: Runge-Kutta Scheme

\[
\partial_t u = D_t (u)
\]

Compute solution using

\[
u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j k_j
\]

Using right-hand side evaluations

\[
k_j = D_t \left( u^n + \Delta t \sum_{l=1}^{s} a_{jl} k_l \right), \quad j = 1, \ldots, s.
\]

with \( s \) the number of steps, \( \Delta t \) the time step size, and \( a_{jl}, b_j \) coefficients for evaluations/update at times \( c_j \).
**Butcher tableau**

\[
\begin{array}{c|ccc}
 c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\
 c_2 & a_{21} & a_{22} & \ldots & a_{2s} \\
 \vdots & \vdots & \vdots & \ddots & \vdots \\
 c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\
\end{array}
\]

- \(c_i\) evaluating times
- \(a_{jl}\) coefficients for computation of evaluations \(k_j\)
- \(b_j\) coefficients for computation of next time step \(u^{n+1}\)

If \(A\) is lower diagonal matrix \(\Rightarrow\) explicit scheme
### Diagonally Implicit Runge Kutta (DIRK) method

Upper diagonal matrix is zero.
Only diagonal entries are implicit.
Implicit-explicit RK methods, [Pareschi, Russo, 2005]

Idea: Split ODE into stiff and non-stiff parts

\[ \partial_t u = -\frac{1}{\epsilon} S(u) - \partial_x F(u) \]

\[ \partial_t u = D_t^{IM} (u) + D_t^{EX} (u) \]

\[ u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} \tilde{b}_j D_t^{EX} (u_j) + \Delta t \sum_{j=1}^{s} b_j D_t^{IM} (u_j) \]

Using intermediate values

\[ u_l = u^n + \Delta t \sum_{j=1}^{l-1} \tilde{a}_{lj} D_t^{EX} (u_j) + \Delta t \sum_{j=1}^{s} a_{lj} D_t^{IM} (u_j) \]
Implicit-explicit (IMEX) methods

\[ u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} \tilde{b}_j D_{t}^{EX} (u_j) + \Delta t \sum_{j=1}^{s} b_j D_{t}^{IM} (u_j) \]

\[ u_l = u^n + \Delta t \sum_{j=1}^{l-1} \tilde{a}_{lj} D_{t}^{EX} (u_j) + \Delta t \sum_{j=1}^{s} a_{lj} D_{t}^{IM} (u_j) \]

Use two Butcher tableaus

\[
\begin{array}{c|c}
\tilde{c} & \tilde{A} \\
\hline
\tilde{b} & \tilde{b}^T
\end{array}
\]

\[
\begin{array}{c|c}
c & A \\
\hline
b & b^T
\end{array}
\]

Implicit scheme is usually a DIRK scheme.
Asymptotic Preserving (AP) property [JIN, 1999]

\[ \partial_t u + A_M \partial_x u = -\frac{1}{\epsilon} S(u) \] (1)

System depends on parameter \( \epsilon \), but for \( \epsilon \to 0 \) the system converges to a limit equation (e.g. Euler equation).

**Definition: Asymptotic-Preserving (AP)**

A scheme for problem (1) with discretization parameter \( \Delta t \) is called **Asymptotic Preserving** (AP) if its stability requirement on \( \Delta t \) is independent of \( \epsilon \) and its limit \( \epsilon \to 0 \) is consistent with the limit solution.

Usually, small \( \epsilon \) makes equation stiffer
\( \Rightarrow \) smaller \( \Delta t \) is needed for explicit schemes \( \Rightarrow \) not AP.
Projective Integration for Moment Equations

joint work with
Giovanni Samaey, KU Leuven
Projective Integration (PI), [Kevrekidis, 2003]

**Idea:**
1. Perform $K + 1$ small time steps $\delta t$ first to damp fast modes.
2. Then extrapolate derivative with one large step $\Delta t$.

0. initialize: $u^{n,0} = u^n$
1. for $k = 0, 1, \ldots, K$:
   
   $u^{n,k+1} = u^{n,k} + \delta t \, D_t (u^{n,k})$
2. extrapolate:
   
   $u^{n+1} = u^{n,K+1} + (\Delta t - (K + 1)\delta t) \cdot \frac{u^{n,K+1} - u^{n,K}}{\delta t}$
Projective Integration as Runge-Kutta method

Standard definition:

\[
\begin{align*}
    u^{n,1} &= u^{n,0} + \delta t D_t (u^{n,0}) \\
    u^{n,2} &= u^{n,1} + \delta t D_t (u^{n,1}) \\
    & \vdots \\
    u^{n,K+1} &= u^{n,K} + \delta t D_t (u^{n,K}) \\
    u^{n+1} &= u^{n,K+1} + (\Delta t - (K + 1) \delta t) \cdot \frac{u^{n,K+1} - u^{n,K}}{\delta t}
\end{align*}
\]

Note

\[
\frac{u^{n,j+1} - u^{n,j}}{\delta t} = D_t (u^{n,j}) = k_j
\]

Write this as Runge-Kutta scheme with evaluations \( k_j \)
Projective Integration as Runge-Kutta method

\[
\begin{align*}
k_0 &= D_t (u^{n,0}) \\
k_1 &= D_t (u^{n,1}) = D_t (u^{n,0} + \delta t k_0) \\
k_2 &= D_t (u^{n,2}) = D_t (u^{n,0} + \delta t k_0 + \delta t k_1) \\
& \vdots \\
k_K &= D_t (u^{n,K}) = D_t \left( u^{n,0} + \delta t \sum_{j=0}^{K-1} k_j \right) \\
\end{align*}
\]

\[
\begin{align*}
u^{n+1} &= u^{n,0} + \Delta t \left( \sum_{j=0}^{K-1} \frac{\delta t}{\Delta t} k_j + \frac{\Delta t - K \delta t}{\Delta t} k_K \right)
\end{align*}
\]
Projective Integration as Runge-Kutta method

\[ k_l = D_t \left( u^{n,l} \right) = D_t \left( u^{n,0} + \Delta t \sum_{j=0}^{l-1} \frac{\delta t}{\Delta t} k_j \right) \]

\[ u^{n+1} = u^{n,0} + \Delta t \left( \sum_{j=0}^{K-1} \frac{\delta t}{\Delta t} k_j + \frac{\Delta t - K \delta t}{\Delta t} k_K \right) \]

Write Projective Integration in Butcher tableau

\[
\begin{array}{c|ccc}
0 & 0 & 0 \\
1 \cdot \frac{\delta t}{\Delta t} & \frac{\delta t}{\Delta t} & 0 \\
\vdots & \vdots & \ddots & \ddots \\
K \cdot \frac{\delta t}{\Delta t} & \frac{\delta t}{\Delta t} & \cdots & \frac{\delta t}{\Delta t} & 0 \\
\frac{\delta t}{\Delta t} & \frac{\delta t}{\Delta t} & \cdots & \frac{\delta t}{\Delta t} & 1 - K \frac{\delta t}{\Delta t} \\
\end{array}
\]

Projective integration does \( K + 1 \) small FE steps and one extrapolation
Projective RK scheme (PRK) [LAFITTE et al., 2017]

Use standard RK scheme

\[ \frac{c}{b^T} \begin{bmatrix} A \\ b \end{bmatrix} \]

\[ u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j k_j \]

\[ k_j = D_t \left( u^n + \Delta t \sum_{l=1}^{s} a_{jl} k_l \right), \quad j = 1, \ldots, s \]

Replace each time derivative \( k_l \) by an inner integrator and a time derivative estimate

\[ u_{i}^{n,k+1} = u_{i}^{n,K} + \delta t D_t \left( u_{i}^{n,K} \right), \quad 0 \leq k \leq K \]

\[ k_l = \frac{u_{i}^{n,K+1} - u_{i}^{n,K}}{\delta t} = D_t \left( u_{i}^{n,K} \right) \]
Stability of Projective Integration

Dahlquist test equation

\[ \partial_t u = \lambda u, \quad \lambda < 0 \]

\[ u^{k+1} = \tau (\lambda \delta t) u^k \]

Amplification factor \( \tau (\lambda \delta t) \).

Stability requires

\[ |\tau (\lambda \delta t)| \leq 1 \]

Forward Euler: \( u^{n+1} = u^n + \Delta t \lambda u^n = (1 + \Delta t \lambda) u^n \)

\[ \Rightarrow \tau^{FE} = (1 + \Delta t \lambda) \]
Stability of Projective Integration

Stability region (PFE): $\mathcal{D}^{PFE} = D \left( 1 - \frac{\delta t}{\Delta t}, \frac{\delta t}{\Delta t} \right) \cup D \left( 0, \left( \frac{\delta t}{\Delta t} \right)^{1/K} \right)$

- First part corresponds to quickly damped modes
- Second part corresponds to slowly decaying modes

Choosing $\delta t$ properly leads to accurate solution of slow modes while maintaining stability of fast modes.
If inner integrator is stable PRK parameters only need to fulfill standard RK stability conditions.

\[
\begin{bmatrix}
c \\
A \\
b^T
\end{bmatrix}
\]

Stability regions of lower-order methods are contained within those of higher-order methods.

Parameters for PFE will be valid for PRK.
Projective Integration does not have unlimited stability w.r.t. \( \lambda \).

For larger \( \lambda \), more inner time steps \( K \) are necessary

\[
\text{for } k = 0, 1, \ldots, K : \quad u^{n,k+1} = u^{n,k} + \delta t D(u^{n,k})
\]

Equivalently, smaller extrapolation steps \( \Delta t \) could be chosen

\[
u^{n+1} = u^{n,K+1} + (\Delta t - (K + 1)\delta t) \cdot \frac{u^{n,K+1} - u^{n,K}}{\delta t}
\]

Still speedup w.r.t. standard Forward Euler method.
Stability of Projective Integration, [Kevrekidis, 2014]

Projective Integration does not have unlimited stability w.r.t. $\lambda$.

For larger $\lambda$, more inner time steps $K$ are necessary

$$\text{for } k = 0, 1, \ldots, K : \quad u^{n,k+1} = u^{n,k} + \delta t D(u^{n,k})$$

Equivalently, smaller extrapolation steps $\Delta t$ could be chosen

$$u^{n+1} = u^{n,K+1} + (\Delta t - (K + 1)\delta t) \cdot \frac{u^{n,K+1} - u^{n,K}}{\delta t}$$

Still speedup w.r.t. standard Forward Euler method.

For kinetic equations, we can still get unlimited stability.
Parameter choice, [Lafitte et al., 2017]

\[
\partial_t f^\epsilon + \frac{v}{\epsilon^\gamma} \partial_x f^\epsilon = \frac{Q(f^\epsilon)}{\epsilon^{\gamma+1}}
\]

\(\gamma = 0\) hydrodynamic scaling,
\(\gamma = 1\) diffusive scaling

Parameter choice

- \(\Delta t = \mathcal{O}(\Delta x)\) for hydrodynamic limit
- \(\delta t = \mathcal{O}(\epsilon)\) for hydrodynamic limit
- \(\Delta t = \mathcal{O}(\Delta x^2)\) for diffusive limit
- \(\delta t = \mathcal{O}(\epsilon^2)\) for diffusive limit
- \(K\) is small number, typically \(K \leq 3\)
Hyperbolic scaling, DVM, linearized BGK, third-order upwind discretization

\[ \delta t = \epsilon, \]

\[ K \geq 2, \]

\[ \Delta t \leq \left( \frac{3\Delta x}{4c_0}, \frac{3\Delta x}{8} \right) \]

where \( c_0 \) is related to the DVM discretization.

\( \Delta t \) might be larger, this is only an estimate.
Projective Integration

Properties of PI

- Small number of inner steps sufficient, $K$ independent of $\epsilon$
- Inner and outer integrators can be RK schemes $\Rightarrow$ high-order
- Works best for spectral gap
Moment models are ideally suited for Projective Integration.
No classical spectral gap

Discrete values for $\epsilon$

Continuous $\epsilon \sim \rho(t, x)$

- adjust number of inner time steps: $K = O(\log(1/\epsilon))$
- prevent stability region split $\Rightarrow [0,1]$-stable method $\Rightarrow \Delta t$ limited by $\epsilon$
- **Telescopic Projective Integration (TPI)**
Idea:
Nested Projective Integration
Integrators at different levels

**Innermost integrator**
- needs to capture fastest components and damp them
- higher-order methods result in stability restriction for other levels
- simply use Forward Euler (FE)

**Projective integrators**
- outermost dominates accuracy of the scheme
- simplest version Forward Euler (TPFE)
- higher-order (outermost) Runge Kutta (TPRK)
Parameter selection [Melis, Samaey, 2018]

- number of steps at level $l$: $K_l$
- time step size at level $l$: $\delta t_l$
- extrapolation size at level $l$: $M_l$

Choice based on spectrum of the respective method
Parameter selection

\[ \partial_t u + A_M \partial_x u = -\frac{\omega(x)}{\epsilon} Qu \]

discrete \( \omega \) levels

- Capture each eigenvalue cluster with one level
- Merge adjacent clusters to one
- Possibility to construct a connected stability region, depends on \( \epsilon \)
Parameter selection

\[
\partial_t u + A_M \partial_x u = -\frac{\omega(x)}{\epsilon} Qu
\]

**continuous \( \omega \)**

- Need to construct a connected stability region, depends on \( \epsilon \)
- \([0, 1]\)-stable method
- Number of levels depends on \( \log(1/\epsilon) \)
Summary of Projective Integration

- PRK, TPI extensions
- AP for const relaxation
- Almost AP for continuous relaxation
- TPI speedups w.r.t. FE
## Further work on Projective Integration

### Past work

[Melis, Samaey, 2018], [Lafitte et al., 2017]
- only for DVM models
- only for BGK and linearized Maxwellians

### Next steps
- implement PI for moment models
- extension TPRK and TPI straightforward
- use PI for other models/collision operators
Further work on Projective Integration

### Past work [Melis, Samaey, 2018], [Lafitte et al., 2017]
- only for DVM models
- only for BGK and linearized Maxwellians

### Next steps
- implement PI for moment models
- extension TPRK and TPI straightforward
- use PI for other models/collision operators

Thank you for your attention!
Efficient asymptotic-preserving (AP) schemes for some multiscale kinetic equations,


Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum,

Implicit-Explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxations

A high-order relaxation method with projective integration for solving nonlinear systems of hyperbolic conservation laws,

Telescopic Projective Integration for Linear Kinetic Equations with Multiple Relaxation Times,